Populating a Data Warehouse from Google Sheets Using Python and Airflow

Data Warehouse Setup

1. **Database Creation**:
    - The data warehouse is implemented using PostgreSQL.
    - Open **pgAdmin4** and create a new database.
    - Execute SQL scripts to create the schema, including **fact tables** and **dimension tables**:
        - Schema: **Star Schema** with a modification:
            - The `clickup` data includes a task category (`project meeting`) absent in the `float` data, resulting in null values during fact table population.
            - A separate fact table, `meeting_fact_hours`, was created to handle this scenario:
                - Excludes `role_id` to avoid null values.
                - Contains `meeting_duration` instead of `total_hours_logged`.
        - Key Tables:
            - **dim_client**
            - **dim_date**
            - **dim_project**
            - **dim_role**
            - **dim_task**
            - **fact_hours**
            - **meeting_fact_hours**

Environment Setup

1. **Set Up Python Environment**:
    - Create a new environment:

      Run: conda create --name sora_union_env

    - Install required packages:

      Run: pip install pandas apache-airflow apache-airflow-providers-postgres sqlalchemy python-dotenv

      Run: pip install --user psycopg2

2. **Airflow Installation**
    - **Install Airflow**

      Run: pip install apache-airflow

    - **Initialize Airflow database:**

      Run: airflow db init

    - **Create an admin user:**

      Run: airflow users create --username admin --password admin --firstname First --lastname Last --role Admin --email admin@example.com

    - **Start Airflow Webserver and Scheduler:**

      Run: airflow webserver --port 8080

      Run: airflow scheduler

Python Script Organization

1. **Configuration and Utilities:**
    - `.env` file: Store database credentials and dataset links.
    - `config.py`: Read credentials from `.env` and initialize the SQLAlchemy engine.
    - `create_logger.py`: Define logging utilities to track ETL runs.
2. ETL Modules:
    - `etl_utilities.py`:
        - Utilities for reading dimension tables, merging data, and breaking DataFrames into smaller chunks.
    - `extract_tools.py`:
        - Methods for extracting data from Google Sheets links.
    - `extract.py`:
        - Implements extraction logic to load data into Pandas DataFrames.
    - `transform_tools.py`:
        - Class `Transform` for renaming columns, handling nulls, and data type conversions.
    - `transform.py`:
        - Applies transformations using the `Transform` class.
    - `load_tools.py`:
        - Class `Load` for writing to PostgreSQL dimension and fact tables.
    - `load.py`:
        - Handles data loading into the data warehouse.
    - `etl_run.py`:

        - Runs the entire pipeline.
    - `Etl_dag.py`:
        - Runs the entire pipeline using Airflow.
3. **Folder Structure:**
    - **Config_files/:** `.env` file, `config.py`
    - **etl_utilities/:** `etl_utilities.py`
    - **extraction_files/:** `extract.py`, `extract_tools.py`
    - **loading_files/:** `load.py`, `load_tools.py`
    - **transformation_files/:** `transform.py`, `transform_tools.py`
    - `Create_logger.py`
    - `Etl_dag.py`
    - `etl_run.py`

Workflow Overview

1. Extraction

- Extract `clickup` **and** `float` data from Google Sheets into Pandas DataFrames.

2. Transformation

- **Column Standardization:**
    - Append column suffixes (e.g., `name`) for schema alignment.
- **Data Cleaning:**
    - Remove duplicates.
    - Handle null values by exclusion or imputation.
- **Date Conversion:**
    - Convert string dates to datetime objects.
- **Dimension Tables:**
    - Create dimension tables (e.g., `dim_task`) by merging and concatenating data from `clickup` **and** `float`**.**
    - Write transformed DataFrames into dimension tables.
- **Fact Tables:**
    - Create `fact_hours` **and** `meeting_fact_hours` tables:
        - Populate using transformed DataFrames and dimension table IDs.
        - Ensure schema consistency with the warehouse.

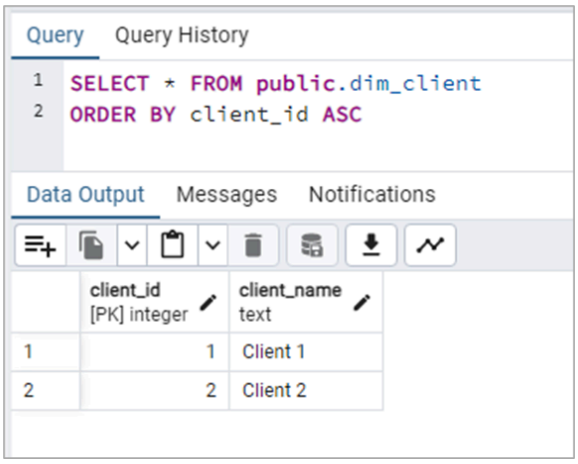
(**Note:** Tthere are two Fact Tables. The Main Fact Table is `fact_hours`. The second Fact Table, `meeting_fact_hours` is a special Fact Table created because of data inconsistency in the `task` columns of the `clickup` and `float` data. The `clickup` data contains **"project meeting"** as an extra value in the `task` column but this extra value cannot be found in the `float` data. Thus, the `meeting_fact_hours` is to cover for this inconsistency. Thus, we have two star schemas: `fact_hours` and the already mentioned dimension tables; and `meeting_fact_hours` and the same dimension tables except the dim_task dimension table)

3. Loading

- **Write data to the warehouse:**
  - **Dimension tables: Use `write_to_dim_table` function to specify columns.**
  - **Fact tables:**
    - **Read written dimension tables and append their IDs to fact DataFrames.**
    - **Write final fact tables to the warehouse.**

Resulting Tables in the DataWarehouse

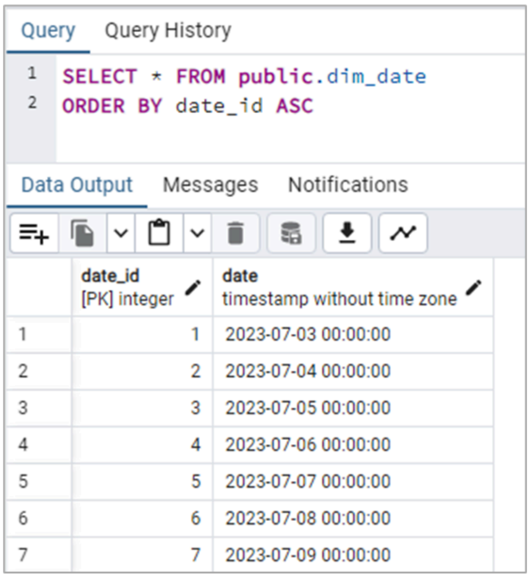
Dim_client Table



```
SELECT * FROM public.dim_client
ORDER BY client_id ASC
```

	client_id [PK] integer	client_name text
1	1	Client 1
2	2	Client 2

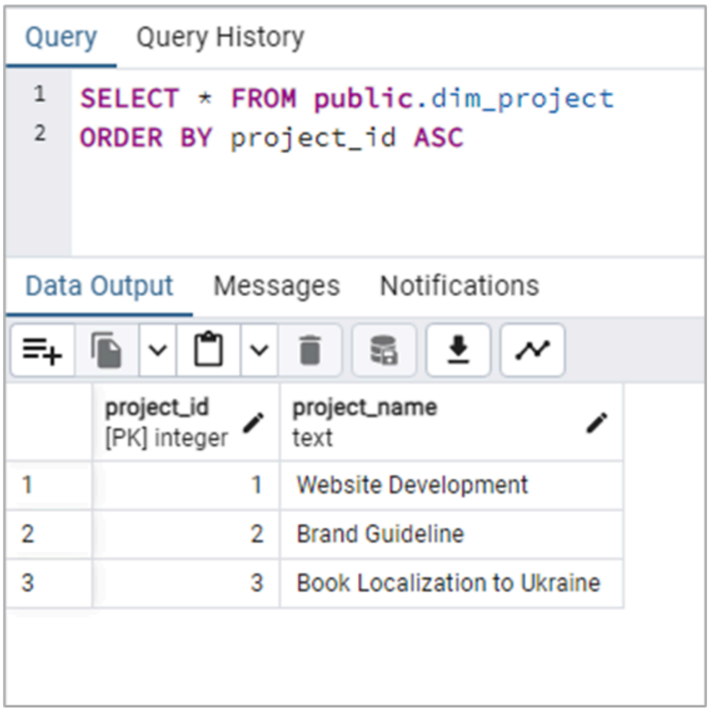
Dim_date Table



```
SELECT * FROM public.dim_date
ORDER BY date_id ASC
```

	date_id [PK] integer	date timestamp without time zone
1	1	2023-07-03 00:00:00
2	2	2023-07-04 00:00:00
3	3	2023-07-05 00:00:00
4	4	2023-07-06 00:00:00
5	5	2023-07-07 00:00:00
6	6	2023-07-08 00:00:00
7	7	2023-07-09 00:00:00

Dim_project Table



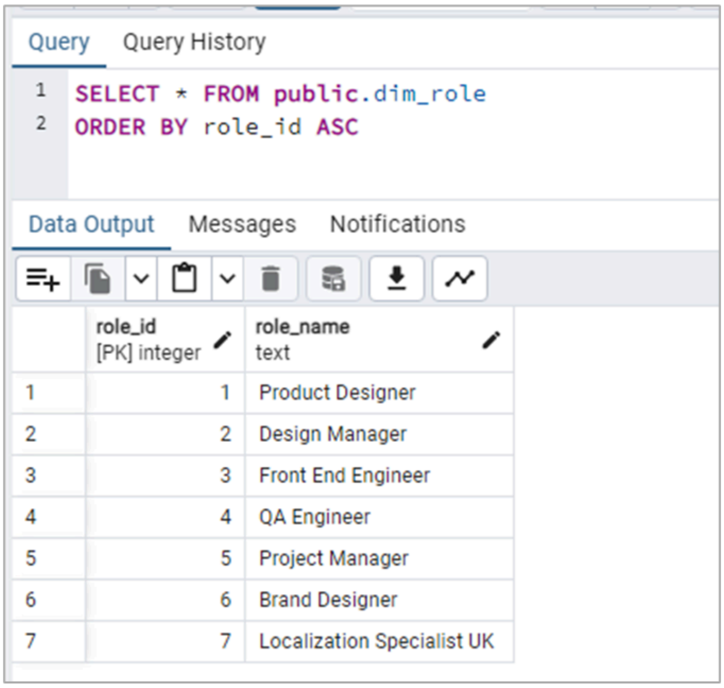
Query | Query History

```sql
SELECT * FROM public.dim_project
ORDER BY project_id ASC
```

Data Output | Messages | Notifications

	project_id [PK] integer	project_name text
1	1	Website Development
2	2	Brand Guideline
3	3	Book Localization to Ukraine

Dim_role Table



Query | Query History

```sql
SELECT * FROM public.dim_role
ORDER BY role_id ASC
```

Data Output | Messages | Notifications

	role_id [PK] integer	role_name text
1	1	Product Designer
2	2	Design Manager
3	3	Front End Engineer
4	4	QA Engineer
5	5	Project Manager
6	6	Brand Designer
7	7	Localization Specialist UK

Dim_task Table

Query | Query History

```
SELECT * FROM public.dim_task
ORDER BY task_id ASC
```

Data Output | Messages | Notifications

	task_id [PK] integer	task_name text
1	1	Design
2	2	Project Meeting
3	3	Development
4	4	Testing
5	5	Management
6	6	Localization

Fact_Hours Table

Query | Query History

```
SELECT * FROM public.fact_hours
ORDER BY fact_id ASC
```

Data Output | Messages | Notifications

	fact_id [PK] integer	client_id integer	project_id integer	role_id integer	task_id integer	date_id integer	total_hours_logged numeric	is_billable text
1	1	1	1	1	1	1	6.5	Yes
2	2	1	1	1	1	2	6.5	Yes
3	3	1	1	1	1	3	6.0	Yes
4	4	1	1	1	1	4	7.0	Yes
5	5	1	1	1	1	5	7.0	Yes
6	6	1	1	1	1	6	7.5	Yes
7	7	1	1	1	1	7	7.0	Yes
8	8	1	1	1	1	8	7.0	Yes
9	9	1	1	1	1	9	7.0	Yes
10	10	1	1	1	1	10	7.0	Yes

Meeting_fact_hours Table

Query   Query History

```
SELECT * FROM public.meeting_fact_hours
ORDER BY meeting_id ASC
```

Data Output   Messages   Notifications

	meeting_id [PK] integer	client_id integer	project_id integer	task_id integer	date_id integer	meeting_duration numeric	is_billable text
1	1	1	1	2	5	1.5	No
2	2	1	1	2	8	1.0	No
3	3	1	1	2	20	1.0	No
4	4	1	1	2	8	2.0	No
5	5	1	1	2	9	1.0	No
6	6	1	1	2	26	2.0	No
7	7	1	1	2	32	1.0	No
8	8	1	1	2	33	0.5	No